Servidor FTP com Node.js

Objetivo: Após o guia anterior, onde você criou um servidor web, agora vamos dar um passo adiante. Você aprenderá o que é um servidor FTP e construirá o seu próprio para transferir arquivos entre seu computador e o servidor, tudo isso usando o poder do Node.js.

Antes de Começar: A Analogia do Armazém Digital

Se o servidor web que criamos antes era como a **vitrine de uma loja** (mostrando informações ao público), um servidor FTP (File Transfer Protocol) é como o **armazém dos fundos** dessa loja.

- **O Armazém (Servidor FTP):** É um espaço organizado para guardar e gerenciar "produtos" (arquivos).
- **A Chave (Login e Senha):** Você precisa de uma chave de acesso para entrar. Nem todo mundo pode entrar no armazém, apenas pessoas autorizadas.
- **As Ações (Upload/Download):** Uma vez dentro, você pode colocar novos produtos nas prateleiras (**upload**) ou pegar produtos existentes para levar embora (**download**).

Em resumo, enquanto o HTTP (servidor web) é excelente para *visualizar* conteúdo, o FTP é o protocolo clássico para *transferir e gerenciar* arquivos de forma eficiente.

Pré-requisitos: Este tutorial assume que você já completou o guia anterior e tem o **Node.js** e o **NPM** instalados e funcionando em sua máquina.

Parte 1: Preparando o Projeto do Servidor FTP

Vamos começar montando a estrutura do nosso "armazém".

1. **Crie a Pasta do Projeto**: Em um local de sua preferência, crie um novo diretório para este projeto. Vamos chamá-lo de `meu-servidor-ftp`. Navegue para dentro desta pasta usando o terminal.
2. **Inicie o Projeto**: Assim como antes, vamos usar o NPM (nosso "gerente") para criar o arquivo de identidade do projeto, o `package.json`.

   ```
   npm init -y
   ```

3. **Instale a Ferramenta Principal**: Para construir nosso servidor FTP, não usaremos o Express, mas sim uma biblioteca especializada chamada `ftp-srv`. Ela contém todas as ferramentas prontas para lidar com conexões e transferências de arquivos. Vamos instalá-la:

   ```
   npm install ftp-srv
   ```

Parte 2: Construindo e Configurando o Servidor

Agora, vamos escrever o código que dará vida ao nosso servidor.

1. **Crie o Arquivo do Servidor**: Na raiz do seu projeto (`meu-servidor-ftp`), crie um arquivo chamado `server.js`.
2. **Programe o Servidor FTP**: Abra o `server.js` e cole o código abaixo. Os comentários explicam cada passo do processo.

```
// 1. Importa a biblioteca do servidor FTP que acabamos de instalar.
const FtpSrv = require('ftp-srv');

// 2. Define a porta onde nosso servidor vai "ouvir".
// A porta padrão do FTP é a 21, mas ela geralmente exige permissões de administrador.
// Usaremos uma porta alta como 2121 para evitar problemas.
const port = 2121;
const ftpServer = new FtpSrv({
    // Define a URL que será usada para se conectar ao servidor.
    // '127.0.0.1' significa que ele só será acessível no seu próprio computador (localhost).
    url: `ftp://127.0.0.1:${port}`,
    anonymous: false, // Não vamos permitir logins anônimos.
    pasv_url: '127.0.0.1', // Necessário para a maioria dos clientes FTP.
});

// 3. A parte mais importante: Gerenciar o acesso (autenticação).
// Este evento é disparado toda vez que alguém tenta fazer login.
```

```
ftpServer.on('login', (data, resolve, reject) => {
    // 'data' contém o usuário e a senha que a pessoa tentou usar.
    const { username, password } = data;

    // --- LÓGICA DE AUTENTICAÇÃO SIMPLES ---
    // Aqui, verificamos se o usuário e a senha estão corretos.
    // **AVISO:** Em um projeto real, senhas NUNCA devem ser escritas
    // diretamente no código! Isto é apenas para fins de aprendizado.
    if (username === 'aluno' && password === '1234') {
        console.log(`✅ Usuário '${username}' conectou-se com sucesso.`);

        // Se o login for bem-sucedido, 'resolve' diz ao servidor para permitir o acesso.
        // O 'root' define qual pasta este usuário poderá acessar.
        // É como dar a ele a chave para um "box" específico no nosso armazém.
        return resolve({ root: './ftp-files' });
    } else {
        // Se as credenciais estiverem erradas, 'reject' nega o acesso.
        console.log(`❌ Tentativa de login falhou para o usuário '${username}'.`);
        return reject(new Error('Credenciais inválidas. Tente novamente.'));
    }
});

// 4. Inicia o servidor e o coloca para escutar na porta que definimos.
ftpServer.listen().then(() => {
    console.log(`🚀 Servidor FTP está no ar e ouvindo na porta ${port}`);
    console.log(`Use um cliente FTP para se conectar a: ftp://localhost:${port}`);
});
```

Parte 3: Preparando os Arquivos e Executando

Nosso código diz que usuários autorizados terão acesso a uma pasta chamada `ftp-files`. Precisamos criá-la.

1. **Crie a Pasta de Arquivos**: Na raiz do seu projeto, crie uma pasta com o nome exato que definimos no código: `ftp-files`.
2. **Crie um Arquivo de Teste**: Dentro da pasta `ftp-files`, crie um arquivo de texto simples. Chame-o de `boas-vindas.txt` e escreva uma mensagem dentro, como "Se você está lendo isso, o servidor FTP funcionou!".
3. **Inicie o Servidor**: Agora, volte ao seu terminal (que deve estar na pasta `meu-servidor-ftp`) e execute o script:

   ```
   node server.js
   ```

   Você deverá ver a mensagem de que o servidor está no ar.

Parte 4: Conectando-se ao Servidor

Para interagir com um servidor FTP, precisamos de um programa chamado **cliente FTP**. Um dos mais famosos e gratuitos é o **FileZilla**. Se você não o tiver, baixe-o e instale-o.

1. Abra o FileZilla.
2. Na barra de "Conexão Rápida" no topo, preencha os seguintes campos:
   - **Host**: `localhost` (ou `127.0.0.1`)
   - **Nome de usuário**: `aluno` (o que definimos no código)
   - **Senha**: `1234` (a que definimos no código)
   - **Porta**: `2121` (a porta que escolhemos)
3. Clique em **"Conexão Rápida"**.

Se tudo deu certo, você verá uma série de mensagens de status na janela superior e, na janela do lado direito (que representa o "Site Remoto" / Servidor), você verá a pasta `ftp-files` e dentro dela o seu arquivo `boas-vindas.txt`!